

## **AMENDMENTS**

### **In the Claims**

The following is a marked-up version of the claims with the language that is underlined (“\_\_\_”) being added and the language that contains strikethrough (“—”) being deleted:

1. (Currently Amended) A method for managing shared resources in a computer system, comprising:

establishing and registering a plurality of objects in response to requests from hardware or software associated with the computer system, the objects including at least one type, at least one attribute, and a handle;

establishing a plurality of message pool objects, wherein the plurality of message pool objects comprise pools of free messages that can be allocated; ~~and~~

manipulating the plurality of objects to effect processing and exchange of ~~information.~~  
information;

receiving, at a message pool interface, a request by a first task object interface for a message allocation;

allocating a message from the free message pool to the first task object;

sending the message from the first task object interface to a second task object interface;

performing processing by the second task object in response the message receipt;

returning the message to the first task object interface upon completion of processing;

and

returning the message from the first task object interface to the message pool interface.

2. (Previously Presented) The method of claim 1, further comprising:
  - establishing a plurality of task objects;
  - allocating messages from at least one free message pool object in response to requests from one or more task objects, wherein the messages include blocks of information that can be passed to other task objects;
  - exchanging the messages between the plurality of task objects, thereby effecting requests for processing; and
  - returning the messages to the free message pool object upon completion of processing.
3. (Previously Presented) The method of claim 2, further comprising:
  - the plurality of task objects include at least a task type and an interface type, the interface type enabling request and release of messages; and
  - the plurality of message pool objects include at least a pool type and an interface type.
4. (Previously Presented) The method of claim 2, wherein exchanging the messages between the plurality of task objects, thereby effecting requests for processing further comprises at least one of: putting a message to an interface, getting a message from an interface, and waiting for a message to arrive on an interface.
5. (Previously Presented) The method of claim 3, further comprising:
  - receiving, at a message pool interface, a request by a first task object interface for a message allocation;
  - allocating a message from the free message pool to the first task object;
  - sending the message from the first task object interface to a second task object interface;
  - performing processing by the second task object in response the message receipt; and

returning the message to the message pool interface upon completion of processing.

6. (Canceled)
7. (Previously Presented) The method of claim 3, further comprising:  
receiving, at a message pool interface, a request by a first task object interface for a message allocation;  
allocating a message from the free message pool to the first task object;  
sending an arm interrupt message from the first task object interface to a interrupt object interface;  
disabling an interrupt with the arm interrupt message by the interrupt object; and  
returning the message to the first task object interface.
8. (Currently Amended) The method of claim 1, further comprising:  
defining a plurality of top-level tasks from the plurality of objects;  
providing each of the plurality of top-level tasks with a private memory resource; and  
enabling access of the private memory resource to any subtask created by a top-level task.
9. (Previously Presented) The method of claim 1, further comprising:  
allocating a memory space to a parent task;  
establishing at least one subtask to the parent task;  
enabling access of the memory space to the at least one subtask; and  
preventing access of the memory space to tasks not associated with the parent task.
10. (Previously Presented) The method of claim 9, further comprising:

allocating a memory space to a subtask; and  
preventing access of the memory space to a parent task of the subtask.

11. (Previously Presented) The method of claim 1, further comprising:  
establishing an object instance for each of the plurality of objects; and  
establishing an object handle for each object instance, the object handle referencing a data structure used to implement the object instance.
12. (Previously Presented) The method of claim 11, wherein the object handle is a pointer value.
13. (Previously Presented) The method of claim 11, further comprising:  
establishing at least one derived object type, based upon the object instance;  
establishing object attributes for the at least one derived object type; and  
accessing any established object attributes with the object handle.
14. (Previously Presented) The method of claim 13, further comprising appending data structures associated with the at least one derived object type to the data structure used to implement the object instance.
15. (Currently Amended) The method of claim 1, further comprising:  
establishing an object instance for each of the plurality of objects;  
establishing at least one derived object type, based upon the object instance;  
establishing object attributes for the at least one derived object type; and  
establishing an object handle for each derived object type, the object handles referencing a data structure used to implement the object instance.

16. (Previously Presented) The method of claim 1, further comprising:  
organizing the plurality of objects as files in a global file system,  
wherein files in the system contain references to objects in memory; and referencing  
each of the plurality of objects in relation to a plurality of top level object types.
17. (Previously Presented) The method of claim 16, wherein the plurality of top level object  
types include tasks, interfaces, pools, mutexes, semaphores, interrupts, and memory.
18. (Currently Amended) A computer-readable medium incorporating one or more  
instructions for managing shared resources in a computer system, comprising:  
one or more instructions for establishing and registering a plurality of objects in response  
to requests from hardware or software associated with the computer system, the objects  
including at least one type, at least one attribute, a handle;  
one or more instructions for establishing a plurality of message pool objects, wherein the  
plurality of message pool objects comprise pools of free messages that can be allocated; and  
one or more instructions for manipulating the plurality of objects to effect processing and  
exchange of ~~information~~: information;  
one or more instructions for receiving, at a message pool interface, a request by a first  
task object interface for a message allocation;  
one or more instructions for allocating a message from the free message pool to the first  
task object;  
one or more instructions for sending the message from the first task object interface to a  
second task object interface;  
one or more instructions for performing processing by the second task object in  
response the message receipt;

one or more instructions for returning the message to the first task object interface upon completion of processing; and

one or more instructions for returning the message from the first task object interface to the message pool interface.

19. (Previously Presented) The computer-readable medium of claim 18, further comprising:  
one or more instructions for establishing a plurality of task objects;  
one or more instructions for allocating messages from at least one free message pool object in response to requests from one or more task objects, wherein the messages include blocks of information that can be passed to other task objects;  
one or more instructions for exchanging the messages between the plurality of task objects, thereby effecting requests for processing; and  
one or more instructions for returning the messages to the free message pool object upon completion of processing.

20. (Previously Presented) The computer-readable medium of claim 19, further comprising:  
one or more instructions for the plurality of task objects include at least a task type and an interface type, the interface type enabling request and release of messages; and  
the plurality of message pool objects include at least a pool type and an interface type.

21. (Previously Presented) The computer-readable medium of claim 19, wherein the one or more instructions for exchanging the messages between the plurality of task objects, thereby effecting requests for processing further comprise at least one of: one or more instructions for putting a message to an interface, getting a message from an interface, and waiting for a message to arrive on an interface.

22. (Previously Presented) The computer-readable medium of claim 20, further comprising:

- one or more instructions for receiving, at a message pool interface, a request by a first task object interface for a message allocation;
- one or more instructions for allocating a message from the free message pool to the first task object;
- one or more instructions for sending the message from the first task object interface to a second task object interface;
- one or more instructions for performing processing by the second task object in response the message receipt; and
- one or more instructions for returning the message to the message pool interface upon completion of processing.

23. (Canceled)

24. (Previously Presented) The computer-readable medium of claim 20, further comprising:

- one or more instructions for receiving, at a message pool interface, a request by a first task object interface for a message allocation;
- one or more instructions for allocating a message from the free message pool to the first task object;
- one or more instructions for sending an arm interrupt message from the first task object interface to a interrupt object interface;
- one or more instructions for disabling an interrupt with the arm interrupt message by the interrupt object; and
- one or more instructions for returning the message to the first task object interface.

25. (Previously Presented) The computer-readable medium of claim 18, further comprising:

one or more instructions for defining a plurality of top-level tasks from the plurality of objects;

one or more instructions for providing each of the plurality of top-level tasks with a private memory resource; and

one or more instructions for enabling access of the private memory resource to any subtask created by a top-level task.

26. (Previously Presented) The computer-readable medium of claim 18, further comprising:

one or more instructions for allocating a memory space to a parent task;

one or more instructions for establishing at least one subtask to the parent task;

one or more instructions for enabling access of the memory space to the at least one subtask; and

one or more instructions for preventing access of the memory space to tasks not associated with the parent task.

27. (Previously Presented) The computer-readable medium of claim 26, further comprising:

one or more instructions for allocating a memory space to a subtask; and

one or more instructions for preventing access of the memory space to a parent task of the subtask.

28. (Previously Presented) The computer-readable medium of claim 18, further comprising:

one or more instructions for establishing an object instance for each of the plurality of objects; and one or more instructions for establishing an object handle for each object instance, the object handle referencing a data structure used to implement the object instance.



29. (Previously Presented) The computer-readable medium of claim 28, wherein the object handle is a pointer value.

30. (Previously Presented) The computer-readable medium of claim 28, further comprising:  
one or more instructions for establishing at least one derived object type, based upon the object instance;

one or more instructions for establishing object attributes for the at least one derived object type; and

one or more instructions for accessing any established object attributes with the object handle.

31. (Previously Presented) The computer-readable medium of claim 30, further comprising one or more instructions for appending data structures associated with the at least one derived object type to the data structure used to implement the object instance.

32. (Currently Amended) The computer-readable medium of claim 18, further comprising:  
one or more instructions for establishing an object instance for each of the plurality of objects;

one or more instructions for establishing at least one derived object type, based upon the object instance;

one or more instructions for establishing object attributes for the at least one derived object type; and

one or more instructions for establishing an object handle for each derived object type, the object handles referencing a data structure used to implement the object instance.

33. (Previously Presented) The computer-readable medium of claim 18, further comprising:

one or more instructions for organizing the plurality of objects as files in a global file system, wherein files in the system contain references to objects in memory; and

one or more instructions for referencing each of the plurality of objects in relation to a plurality of top level object types.

34. (Previously Presented) The computer-readable medium of claim 33, wherein the plurality of top level object types include tasks, interfaces, pools, mutexes, semaphores, interrupts, and memory.